Programmable Logic Card User Guide

August 7, 2015

1 Overview

ASI's Programmable Logic Card (PLC) is a field-programmable card for digital logic, sort of a mini-FPGA inside the Tiger controller. In the Tiger controller, it acts as a single axis device (by default, axis E) and occupies one Tiger address (e.g. 36 in a diSPIM TG-16). Its original use is to enable multi-laser control for diSPIM, but it is designed to be very flexible.

The PLC has 8 accessible I/Os in the form of BNC connectors on the front panel, as well as connections to the Tiger backplane TTL lines (used in diSPIM for camera triggers and inter-card communication). The PLC also has an array of 16 programmable logic cells[1]; each logic cell has up to 4 inputs[2] and a single output. Inputs to each cell are selectable from the outputs of any logic cell, BNC inputs, and the Tiger backplane. Cell outputs can be routed to the BNC connectors or to the Tiger backplane. The core of each cell is a user-programmable logic element; possibilities include combinatorial logic, flip flops, programmable one-shot and delay elements. As of February 2015 there are 15 different types of cells available.

The PLC periodically runs an evaluation cycle. During each evaluation cycle, the card does the following:

1. Updates external I/Os with the outputs from the prior evaluation cycle[3]
2. Samples all inputs synchronously[4]
3. Computes the new outputs one cell at a time, starting with cell 1 and proceeding to cell 16. The computed outputs won't be available on the external I/Os until the start of the next evaluation cycle. [5]

The evaluation cycle is triggered by a clock signal. By default, the evaluation clock comes from an 4 kHz source on the PLC. It is also possible for the clock to be provided by another Tiger card for synchronization purposes, e.g. for diSPIM the micro-mirror card is the source of the signals being output by the PLC (e.g. camera triggers) so the micro-mirror is the master and its 4 kHz clock is used to trigger PLC evaluations. Finally, the user can provide a clock on the first BNC input.

Note that there is an delay of one evaluation cycle because the outputs are updated at the start of the evaluation cycle; this is an intentional design choice so that the output timing is deterministic. Thus, an input signal with arbitrary phase relative to the evaluation clock is output with a delay between 1 and 2 times the period of the evaluation cycle clock.

2 Programming

The configuration of each cell can be changed by sending serial commands to the card. This can be done by either setting the entire card configuration to one of several presets or by sequentially setting the configuration of

[1] The number of logic cells is easily extendable, but then longer evaluation times are required. It may be beneficial to let the user say how many logic cells are used.

[2] It is easily extendable to 6 inputs, but 4 is enough for logic cells we have now.

[3] not strictly simultaneous but within 0.3 μs of the trigger (verified with internal triggering, should do again with external)

[4] again, not strictly synchronously but within 0.5 μs of the trigger

[5] If cell 2 uses the output of cell 1, there will be no apparent delay, but if cell 1 uses the output of cell 2 then it won't be updated until the next evaluation cycle.

each cell. The configuration of the entire card is saved by executing a "save settings" operation, and need not be reprogrammed again until a different logic function is needed. The use of each physical connector is also user-programmed; each connector can be used as an digital input or mapped to reflect the output of any desired logic cell. Besides the 8 faceplate connectors, there are also 8 backplane connectors for inter-card communication in the controller (e.g. used in the diSPIM).[6]

2.1 Programming Logic Cells

To program the logic cells:

1. Move the axis position to the cell, e.g. (**`M E=10`**) specifies that the programming commands **`CCA Y Z F`** and **`CCB`** will be applied to the cell at position 10. The current position can be queried using **`W`**.

2. Execute **`[Addr#]CCA Y=<code>`** to select the type of cell. See Table 1 for cell types and codes. Setting the cell type (even to the exact same value) clears the previously-selected cell configuration, input selections, and state.

3. Execute **`[Addr#]CCA Z=<code>`** to set the cell configuration if applicable (see Table 1). This is a 16-bit code that defines how the cell behaves, and its interpretation varies by cell type. Setting the cell configuration clears the state (only applies to one-shots and delay cells).

4. Execute **`[Addr#]CCB X=<input #1> Y=<input #2> Z=<input #3> F=<input #4>`** to set cell inputs, where the input address is from Table 2. Inputs default to 0, or logic low. Note that the logical inversion of any address is easily obtained by simply adding 64 to the address. To obtain a rising or falling edge signal, add 128 or 192 to the address respectively. When an address is assigned to an edge-sensitive input as marked in italics in Table 1, the firmware automatically converts the address to the range 128–255 if it is specified by the user in the range 0–127 (when such an input address is queried the value between 128 and 255 is returned). It is possible to combine the edge signals using combinatorial logic combine signals to feed to edge-sensitive inputs; e.g. to clock a flip flop when the output of cell 1 changes (either rising or falling edge) define a 2-input OR with input addresses 129 and 193 ($193 = 1 + 128 + 64$) and use the OR gate's output to clock the flip flop.[7]

The remainder of this section discusses available cell types.

2.1.1 Logic gates

The predefined cell types include some common boolean logic gates for convenience, even though lookup tables could also be used. Two-input AND, OR, and XOR gates are provided, along with four-input AND and OR gates. A 3-input AND can be created using a 4-input cell and setting the unused input to address 64 (logic 1), and similarly a 3-input OR by setting the unused inputs to 0.

2.1.2 Lookup tables (LUT)

Lookup tables (LUTs) are used to implement arbitrary boolean logic. In the case of a 4-input LUT, the eponymous lookup table code comprises 16 bits corresponding to the digital outputs of the 16 possible combinations of the 4 digital inputs. It is programed as the **`CCA Z`** code which ranges between 0 and 65535, where the LSB (bit 0) corresponds to all inputs 0 (logic low) and the MSB (bit 15) corresponds to all inputs 1 (logic high). Bit 1 of the lookup table code is the output when Input #1 is high and the others low, bit 2 of the code is the output when Input #2 is high and the others low, bit 3 is the output when Inputs #1 and #2 are high and the others are low, etc. The 16-bit lookup table code can be derived as follows:

[6] An additional 8 backplane connectors are reserved but not used at present.
[7] Really the OR gate's address + 128 will be assigned as the flip-flop's input address, but it is equivalent in function.

Cell	Cell Type (**CCA Y**)	Configuration (**CCA Z**)	Input #1 (**CCB X**)	Input #2 (**CCB Y**)	Input #3 (**CCB Z**)	Input #4 (**CCB Z**)	Has State?
constant	0	0 for 0, 1 for 1	-	-	-	-	-
D-flop	1	-	Din	*clock*	reset	preset	Yes
2-input LUT	2	LUT values	A	B	-	-	-
3-input LUT	3	LUT values	A	B	C	-	-
4-input LUT	4	LUT values	A	B	C	D	-
AND2	5	-	A	B	-	-	-
OR2	6	-	A	B	-	-	-
XOR2	7	-	A	B	-	-	-
one-shot (retriggerable)	8	duration (# clock pulses)	*trigger*	*clock*	reset	-	Yes
delay (retriggerable)	9	delay to fire (# clock pulses)	*trigger*	*clock*	reset	-	Yes
AND4	10	-	A	B	C	D	-
OR4	11	-	A	B	C	D	-
Synchronous D-flop	12	-	Din	*clock*	reset	preset	Yes
JK-flop	13	-	J	K	*clock*	-	Yes
one-shot (non-retriggerable)	14	duration (# clock pulses)	*trigger*	*clock*	reset	-	Yes
delay (non-retriggerable)	15	delay to fire (# clock pulses)	*trigger*	*clock*	reset	-	Yes

Table 1: Cell codes for use with CCA and CCB commands (cell type and configuration). *Trigger and clock inputs are edge-sensitive and are rendered in this font*; all other inputs are level-sensitive. If the user sets an edge-sensitive input to address 0–127 then it will automatically be set to the corresponding edge-sensitive address 128–255. Address 192 (or 64) has a rising edge every evaluation cycle.

Address	Meaning	Type
0	always 0/low	constant
1–16	logic cell 1–16	logic cell
17–32	reserved for future use	(expanded cell array)
33–40	front panel connectors 1–8	physical I/O
41–48	Tiger backplane TTL 0–7	physical I/O
49–63	reserved for future use	(expanded I/O)
64–127	0–63 with inversion	
128–191	rising edge on 0–63	
192–255	falling edge on 0–63	

Table 2: Address codes; used with **CCB** command as well as setting source address for physical I/O.

input #1 (**X**)	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
input #2 (**Y**)	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
input #3 (**Z**)	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
input #4 (**F**)	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
output	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
weight	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	64	32	16	8	4	2	1

First determine the desired output (0 or 1) in the penultimate row of the table. This is the binary form of the configuration code. To determine the decimal form used by the **CCA Z** command to program the cell, multiply each the output by the binary weight below and sum the values. For example, with a 4-input lookup table the operation ((X AND Y) OR ($\overline{X}$ AND $\overline{Y}$ AND $\overline{Z}$ AND $\overline{F}$)) is represented by the **CCA Z** code of 1000100010001001 in boolean, or34953 in decimal, and would be programmed by sending the command **CCA Z=34953**. The operation (Z OR F) is represented by the LUT 1111111111110000 in boolean, or 65280 in decimal.

If only 2 or 3 inputs are required it is preferred to use the 2-input or 3-input LUT versions for slightly faster execution. For a 3-input LUT the input #4 (**F**) doesn't matter, and for a 2-input LUT inputs #3 and #4 (**Z** and **F**) don't matter, meaning the maximum **CCA Z** code would be 255 and 15 respectively.

2.1.3 Flip flops

The D flip-flop is a simple latch or 1-bit memory. If its reset input is high then the output is unconditionally set low. If preset is high and reset is low, then the output is high. If both reset and preset are low (the normal state), then the output is the input at the time the last rising edge was seen on the clock input. The output is updated the same evaluation cycle as the clock, i.e. the D-to-Q delay is 0. The reset and preset only require an evaluation cycle to "take," not a clock pulse; i.e. they act like asynchronous reset/presets.

The synchronous D-flop behaves just like the normal D-flop except the reset and preset inputs are only sampled on the rising clock edge, acting as synchronous reset/preset.

The low-level JK-flop is helpful to build some logic structures. It has two inputs, J and K along with a clock input. When the clock input has a rising edge, the output is updated according to its prior state and the J and K input values. The combination J=1 and K=0 sets the output to 1, the combination J=0 and K=1 sets the output to 0, the combination J=K=1 will toggle the output value, and J=K=0 will hold the current state.

2.1.4 One-shot

The output of a one-shot (also known as a monostable multivibrator) goes high for a specified duration upon receiving a rising edge on the trigger input. The output stays high until it has detected the user-specified number of rising edges on the clock input in later evaluation cycles (i.e. the clock input on the evaluation cycle when the trigger has a rising edge is ignored), and then its output goes low again. The number of clock cycles, or duration, is specified as the configuration for this cell using **`CCA Z`** (must be between 0 and 65535). Note that the clock signal is an input; if you wish it to be the same as the clock for the PLC evaluation cycle then set input #2 to address 192 (setting to 64 has the same result). If the duration is set to 0 then the output will never go high. If the duration is set to 1, then the one-shot output goes high on the trigger pulse and low again at the first rising edge on the clock input. The reset input resets the count and forces the output to be 0; it overrides the other inputs.

There are two varieties of one-shots. The standard one is retriggerable, meaning a trigger received while the output is high will cause the duration counter to be reset and the pulse length to be extended. In other words, the output will go low the specified number of clock cycles after the last-received trigger. For the non-retriggerable one-shot, the trigger input is ignored while the output is high, so the duration will always be the same unless it is reset.

2.1.5 Delay

The delay cell is somewhat like the one-shot, except for its output goes high for a single clock cycle (not evaluation cycle[8]) some user-specified time after the initial trigger. The number of clock cycles, or duration, before the output goes high is set by the cell's configuration (**`CCA Z`**, must be between 0 and 65535) . There is a special case if the duration is set to 0: the output will go high immediately and low again on the next rising edge of the clock signal. If the duration is set to 1 then the output will go high on the next rising clock edge and low again on the second rising edge. With duration of 2 then the output goes high on the second rising edge and low again on the third edge, and so forth. If a delay and one-shot cell are both connected to the same trigger and clock signals and have the same configuration (duration) setting then the delay cell's output goes high the same evaluation cycle as the one-shot's output goes low again. Like the one-shot, the clock input is ignored on the evaluation cycle when the trigger's rising edge is received. Also like the one-shot, the reset input resets the count and forces the output to be 0; it overrides the other inputs.

A one-shot with delay can be created by combining a one-shot cell and delay cell, with the delay cell's output connected to the trigger signal of a one-shot cell, both running from the same clock.

The standard delay cell is retriggerable, meaning that the duration counter is reset by receipt of another trigger so the output pulse will come the specified amount of time after the last-received trigger. A non-retriggerable delay cell is also available, which ignores trigger pulses between the initial trigger and the output pulse.

2.2 Configuring Physical I/Os

Physical I/Os include both the front panel BNCs and the TG-1000 backplane lines. Each has two associated registers containing the I/O type and the source address (from which address the output will come).

2.2.1 I/O type: input, open-drain output, or push-pull output

There are three types of I/Os: input, open-drain output, or push-pull output. By default the TG-1000 backplane I/Os are inputs and the front panel BNCs are push-pull outputs.[9]

When an I/O is used as an output, it can either use the open-drain + pull-up resistor configuration or else a push-pull configuration.

For Rev A boards, the push-pull output logic levels are fixed at approximately 0.1 V and 3.2 V (the microcontroller supply voltage) and the source impedance is relatively low (typical value of 150 Ω) but not low

[8] to have the output high for only a single evaluation cycle, connect the delay cell's output to its own reset input

[9] Prior to v3.05 the BNCs were also inputs by default.

enough to drive an accessory with 50 Ω input impedance. For Rev B boards, push-pull outputs go through a separate buffer which source at least 48 mA, sufficient to drive 50 Ωto TTL levels. Also on Rev B boards, the logic high voltage for push-pull is selectable between 5 V (default) or 3.3 V using a jumper on the PCB and the the low logic level is less than 0.1 V.[10]

Open-drain output mode can be used to allow different output logic levels but requires the addition of an external pull-up resistor. Most users will not need to use this mode on the BNC connectors, though it is used to communicate on the TG-1000 backplane. The logic high value should be no more than 5.5 V. The maximum current sink is 100 mA, but it is recommended to be much less (e.g. use 5V with a 5 kΩ pull-up and the current will be 10 mA).

Each I/O has a pull-up or pull-down resistor on the printed circuit board to prevent the line from floating. This resistor acts as the input impedance when the I/O is used as an input.[11] TG-1000 backplane connections have 10 kΩpull-up resistors to 3.3 V and are usually used in open-drain mode to communicate with other Tiger cards. The front panel BNCs each have a 27 kΩresistor to ground (for Rev A boards the value is 10 kΩ). Finally, there is a series resistor between the BNC connectors and internal nodes for protection; the value is 100 Ω for Rev A boards and 43 Ω for Rev B. This series resistor makes it so the output voltage is not strictly 0 when the BNCs are used in open-drain output mode. For Rev B boards, the series resistor makes the push-pull output impedance approximately 50 Ω for impedance matching.

To specify a I/O type, first move the axis pointer to the physical I/O address and then execute a **`[Addr#]CCA Y`** command setting **`Y`** to the I/O type code. The code is **`0`** for input, **`1`** for open-drain output, and **`2`** for push-pull output[12]. For example, to set the first front panel connector to be a push-pull output, do **`M E=33`** followed by **`[Addr#]CCA Y=2`**. This is a separate but distinct use of the **`CCA Y`** command described in Section 2.1; if the current axis position corresponds to a logic cell address then **`CCA Y`** sets the I/O type for the physical I/O, but if the axis position is on a logic cell then the cell type is set.

2.2.2 Physical I/O source address

The source address setting does not matter if the I/O type is an input. If it is an output, the source address specifies from what programmable logic cell or other I/O the output will be taken from.

It is possible to assign the physical I/O source address to be another physical I/O. If the source is an input, note that the data will be sampled when the evaluation cycle is triggered, and also remember the fixed delay of a single clock cycle due to the evaluation cycle structure described in Section 1.

To specify a source address, first move the axis pointer to the physical I/O address and then execute a **`[Addr#]CCA Z`** command setting **`Z`** to the source address. For example, to set the front panel #1 connector to be the output of cell 2, do **`M E=33`** followed by **`[Addr#]CCA Z=2`**. The logical NOT of the signal is easily obtained by simply adding 64 to the source address, e.g. to output the logical complement of cell 2's output value send **`[Addr#]CCA Z=66`** after moving the axis pointer to the correct position. This is a separate but distinct use of the **`CCA Z`** command described in Section 2.1; if the current axis position corresponds to a logic cell address then **`CCA Z`** sets the source address for the physical I/O, but if the axis position is on a logic cell then the cell configuration is set.

2.3 Available Presets

Card presets are programmed using the **`[Addr#]CCA X=<preset #>`** command. Depending on the preset, some or all of the cells will be changed (types, configuration, and input sources) as well as some of the I/Os (I/O types and source addresses). The cells and outputs not used in the preset are not affected. Thus, applying multiple presets can have a "cumulative" effect, as long as they don't overwrite each other. After setting a preset, the cells and outputs can be further modified.

[10] Technically speaking the push-pull buffer is not rated for supply as low as 3.3V but we have never experienced any problems operating it this way.

[11] Strictly speaking the series resistor mentioned later also is part of the input impedance.

[12] For Rev A boards, the micro-controller itself was set to push-pull output and would drive the BNC connector through a resistor. For Rev B and later boards, an additional buffer is used in push-pull mode that is capable of driving a 50 Ω load.

Code	Trigger source	Notes
0	internal 4 kHz clock	default
1	Backplane C7	for diSPIM, from micro-mirror card (P3.0)
2	Backplane C13 (TTL5)	
3	Backplane C14 (TTL7)	
4	BNC input 1	

Table 3: Trigger source codes.

The list of presets available is shown in Tables 4 and 5. Additional presets can be easily added upon request. BNC outputs are push-pull unless otherwise specified.

2.4 Accessing outputs over serial

Most often the binary values of the programmable cells are used as digital signals on the physical connectors. However, the output of the programmable cells can also be read using a serial command. This may be useful, for example, if a counter has been created and high-level software needs to know the counter value. Likewise, the value of the physical I/Os can also be read. Users of these commands should understand that the values may be changing rapidly, and the commands only return a single-time snapshot of the state.

Use the **`[Addr#]RDADC Z?`** command to access the 16-bit unsigned integer corresponding to the output values of the 16 logic cells (addresses 1–16). The output of cell 1 is the LSB and the output of cell 16 is the MSB.

`[Addr#]RDADC X?` returns the value of the front panel I/Os, with LSB corresponding to BNC #1. Floating inputs read as a logic low because board has a pull-down resistor to ground.

`[Addr#]RDADC Y?` returns the value of the Tiger backplane I/Os, with LSB corresponding to TTL0. Floating inputs read as logic high because the board has a pull-up resistor to 3.3 V).

2.5 Changing trigger source

By default the evaluation cycle happens on an internal 4 kHz clock. However, the evaluation cycle can be triggered by hardware interrupt instead. For diSPIM, the micro-mirror card generates trigger signals for cameras and lasers which are controlled by PLC outputs. To maintain synchronization, the PLC trigger source is set to be the 4 kHz clock of the micro-mirror card.

There is an option for the user to provide a clock for the evaluation cycle on the first BNC front panel connector. When changing to this trigger source, BNC input 1 is set to be an input automatically and cannot be changed to an output until the trigger source has been changed.

To change trigger mode, use the **`PM`** command, which is axis-specific, to change the trigger source as desired. For example, **`PM E=1`** would be used for diSPIM with the micro-mirror card master clock is on backplane C7. Table 3 contains a list of trigger source codes.

2.6 Accessing cell state

Flip-flops, one-shots, and delay cells all have state, which refers to the D-flop output (high or low) and the current clock counter value in one-shot and delay elements. Generally the logic cell state is not manipulated directly. Note that cell state is not the same as the binary output value which can be accessed using **`[Addr#]RDADC Z?`**.

Use the **`HOME <axis>`** or **`! <axis>`** command to clear the state of all cells, e.g. **`!E`**.

The command **`CCA F`** can be used to read or write the state of the currently-selected cell according to the axis pointer. For example, if cell 11 contains a D-flop then its state (0 or 1) can be read by first executing

Preset	Description	Cells	BNC Outputs	Firmware	Notes
0	All cells set to constant 0	1–16	-	3.00+	
1	Simulate original SPIM TTL card	-	BNC1–4	3.00+	
2	Set cell 1 to constant 0	1	-	3.00+	diSPIM: not acquiring
3	Set cell 1 to constant 1	1	-	3.00+	diSPIM: acquiring
4	16-bit counter	1–16	-	3.00+	Clock input set to address 192, cell 1 is LSB toggle flop
5	BNC5 selected, turned on by cell 10	-	BNC5–8	3.04+	diSPIM: laser #1 selected
6	BNC6 selected, turned on by cell 10	-	BNC5–8	3.04+	diSPIM: laser #2 selected
7	BNC7 selected, turned on by cell 10	-	BNC5–8	3.04+	diSPIM: laser #3 selected
8	BNC8 selected, turned on by cell 10	-	BNC5–8	3.04+	diSPIM: laser #4 selected
9	BNC5-8 all turned off	-	BNC5–8	3.04+	diSPIM: no laser selected
10	Set cell 8 to constant 0	8	-	3.04+	diSPIM: laser disabled
11	Set cell 8 to constant 1	8	-	3.04+	diSPIM: laser enabled
12	Cell 10 as (TTL1 AND cell 8)	10	-	3.08+	diSPIM: selected laser on
13	BNC4 source as (TTL3 AND (cell 10 OR cell 1)	12	BNC4	3.06+	diSPIM: gated side output
14	diSPIM TTL	1, 8, 10, 12	BNC1–8	3.04+††	Combination of presets 1, 9, 12, 13
15	modulo 4 counter in cells 3 and 4, cell 2 as clock	3, 4	-	3.06+	diSPIM: counter for 4 lasers
16	modulo 3 counter in cells 3 and 4, cell 2 as clock	3–5	-	3.06+	diSPIM: counter for 3 lasers; note also uses cell 5
17	Cell 2 is !TTL1	2	-	3.06+	diSPIM: increment count every slice
18	Cell 2 is !TTL3	2	-	3.06+	diSPIM: increment count every volume (side A first)
19	BNC1–8 source from cells 9–16	-	BNC1–8	3.06+	Useful for testing together w/ preset 4
20	BNC5–8 source from cells 13–16	-	BNC5–8	3.07+	diSPIM: laser source from cells 13–16

Table 4: Card-wide presets 1–20.
†† Prior to v3.09 BNC3 was set to the equivalent of preset 24 as well.

Preset	Description	Cells	BNC Outputs	Firmware	Notes
21	modulo 2 counter in cells 3 and 4, cell 2 as clock (cell 4 always low)	3, 4	-	3.07+	diSPIM: counter for 2 lasers
22	no counter in cells 3 and 4 (both always low)	3, 4	-	3.07+	diSPIM: only 1 channel
23	BNC1–8 source from TTL0–7	-	BNC1–8	3.09+	
24	BNC3 source from cell 1	-	BNC3	3.09+	diSPIM: acquisition indicator
25	BNC3 source from cell 8	-	BNC3	3.09+	diSPIM: shutter indicator
26	Cell 2 is TTL3	2	-	3.10+	diSPIM: increment count every volume (side B first)

Table 5: Card-wide presets 21 and higher.

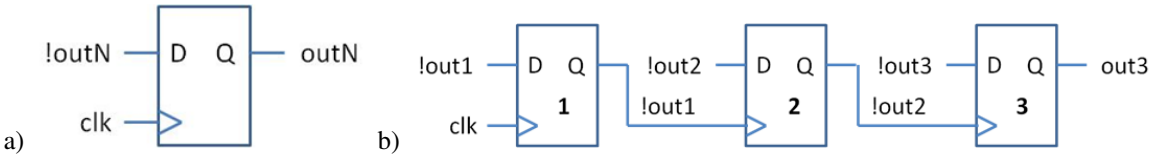

Figure 1: a) Toggle flip flop; b) 3 bit up counter

`M E=11` followed by **`[Addr#]CCA F?`**. Likewise, the D-flop state (and hence output value) could be set high by executing **`M E=11`** followed by **`[Addr#]CCA F=1`**.

2.7 Save card settings

Like other cards, the save settings command (**`[Addr#]SS Z`**) can be sent to the programmable logic card to save all settings in nonvolatile memory where it will be restored when the controller is powered on the next time. Cell states (e.g. flop values) are not remembered, but the following settings are remembered using SS Z:

- logic cell types, configuration, and input assignments
- I/O types and source addresses
- card trigger source

3 Example Logic

3.1 Simple counter

A simple N-bit counter consists of N toggle flops. Each toggle flop is simple in structure, see Figure 1 a). A counter is formed by cascading toggle flops, with the output of each cell acting as the clock for the following cell; Figure 1 b) shows this for a 3-bit up counter. This is knows as a "ripple counter" because the lower order bit has to toggle before the next bit can flip. In digital logic there is a transient time where the observed count will be inaccurate while the outputs are changing. However, in the PLC, as long as each higher-order toggle flop is in a higher-number cell then the counter will act as a synchronous counter, as noted below. To make a down counter, invert the polarity of the clock inputs.

3.2 diSPIM 2 laser control, toggle laser every volume

Say we want to use two lasers with diSPIM where the laser color is changed on alternate volumes. Currently the Tiger outputs are a single laser trigger and a TTL side indicator. For two lasers, we need to route the laser trigger to one laser for the first volume and the second laser on the next volume. This is accomplished by using a toggle flop (1-bit counter) that is toggled after every volume. This would require 3 logic cells, arranged as shown in Figure 2. The BNC connectors on the front of the PLC provide the required additional physical connectors.

3.3 diSPIM 2 laser control, toggle laser every slice

The logic for this is almost identical, except that the toggle counter is connected to the laser trigger as shown in Figure 2 b). Some additional logic could be added to reset the counter between stacks in case an odd number of slices is used.

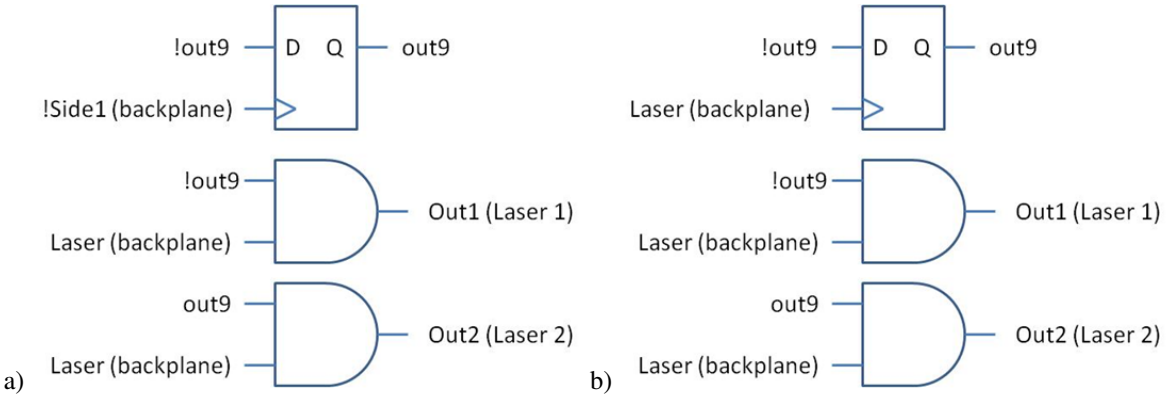

Figure 2: Scheme for 2 laser control, a) alternating lasers every volume and b) alternating lasers every slice.

4 Hardware specifications

4.1 Input/Output Impedance

As described above the BNC impedance values for Rev B boards are as follows: input: 27 kΩ, open-drain output: 43 Ω in series, push-pull output: approximately 50 Ω. The backplane I/Os are implemented using pull-up logic with 10 kΩ resistors on the board. The front panel I/Os are protected with 43 Ωin series and a 27 kΩ to ground (100 Ω and 10 kΩ in Rev A, except for a very few early cards with the 10 kΩresistor to 5 V instead).

4.2 diSPIM backplane

The standard TG-1000 backplane is configured as follows for diSPIM:

PLC Address	Signal	diSPIM use
41	TTL0	Camera Path A
42	TTL1	Laser 0 (laser trigger)
43	TTL2	Camera Path B
44	TTL3	Laser 1 (laser side)
45	TTL4	Piezo Path A
46	TTL5	Trigger Input
47	TTL6	Piezo Path B
48	TTL7	Reserved

Laser 0 and Laser 1 outputs behave according to a firmware setting (**`LED Z`**). This setting is usually changed by the user to match the laser control configuration they are using. The outputs can be set as:

- individual laser on/off for the two paths
- single laser trigger with side switch (Laser 0 is laser on/off for both sides and Laser 1 is high when Path B is active) (default)
- side indication only (Laser 0 high when Path A is active, Laser 1 high when Path B is active)

5 Implementation

Originally it was hoped to run the evaluation cycle at a very fast update rate, e.g. at 100 kHz, so that any propagation time would be negligible. However, that proved unfeasible. Instead we provide a mechanism to slave the outputs to another card's clock.

Need to verify sustained 4 kHz operation independent of logic cell assignment (need to identify worst-case). With all 4-input LUTs it took 0.12 ms to do the evaluation cycle, and with all constants it took 0.07 ms. With 32 programmable cells, all 4-LUTs, it took 0.20 ms, indicating that about 0.04 msis overhead and about 5 μs per cell. Same experiment with constants, the time is about 0.03 msoverhead and 5 μs per cell.

Outputs can be used immediately when evaluating subsequent cells. This makes it possible to implement some functionality with less delay (e.g.the ripple counter would not have any effective "ripple" time, just a single cycle, as long as the higher-order bits are placed in cells evaluated after the lower-order bits). However, note that the logic's behavior can depend on cell ordering. In each evaluation cycle, cell 1 is evaluated first, then cell 2, and so on.

6 Notes

More cell types can easily be added. More presets can easily be added.

Additional programmable cells can also be added. As many as 32 cells may be doable with 0.25 ms clock rate, and we have pre-allocated addresses for that already.